Machine Learning Anomaly-Based Network Intrusion Detection: Experimental Evaluation

Ahmed Ramzi Bahlali, Abdelmalik Bachir

Machine Learning Anomaly-Based Network Intrusion Detection: Experimental Evaluation

Ahmed Ramzi Bahlali[1,2(✉)] and Abdelmalik Bachir[2]

[1] IMATH Laboratory, University of Toulon, La Garde, France
a.ramzi.bahlali@gmail.com
[2] LESIA Laboratory, University of Biskra, Biskra, Algeria
abdelmalik.bachir@univ-biskra.dz

Abstract. The use of Machine Learning (ML) approaches to design anomaly-based network intrusion detection systems (A-NIDS) has been attracting growing interest due to, first, the ability of an A-NIDS to detect unpredictable and previously unseen network attacks, and second, the efficiency and accuracy of ML techniques to classify normal and malicious network traffic compared to other approaches. In this paper, we provide a comprehensive experimental evaluation of various ML approaches including Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), and Artificial Neural Network (ANN), on a recently published benchmark dataset called UNSW-NB15 considering both binary and multi-class classification. Throughout the experiments, we show that ANN is more accurate and has fewer false alarm rates (FARs) compared to other classifiers, which makes Deep Learning (DL) approaches a good candidate compared to shallow learning for future research. Moreover, we conducted our experiments in a way to be served as a benchmark results since our used approaches are trained and tested on the configuration deliberately provided by the authors of UNSW-NB15 dataset for the purpose of direct comparison.

Keywords: Intrusion detection system · Machine learning · Network anomaly detection · Artificiel neural network · Cyber security

1 Introduction

An IDS is a proactive mechanism that monitors inbound and outbound traffic to identify malicious traffic. An IDS could be categorized according to its placement and its detection technique [5,27]. There are mainly two types of IDSs with respect to its placement: Host-Based IDS (HIDS) and Network-Based IDS (NIDS). An HIDS runs on a local host machine and uses system activities to identify malicious behavior locally, whereas NIDS is located at the network entry to protect the internal network from external cyber threats. IDSs can also be classified according to the technique used to detect intrusions into two categories: signature-based

L. Barolli (Ed.): AINA 2023, LNNS 654, pp. 392–403, 2023.
https://doi.org/10.1007/978-3-031-28451-9_34

and anomaly-based IDSs. Signature-based IDSs rely on a preexisting database of well-known attack patterns, and any packet or flow that matches one of those patterns is flagged as malicious [2,15]. This category has a major drawback as it requires continuous manual update of the database of signatures by adding up-to-date attacks and thus cannot detect unseen malicious patterns [2,15]. In contrast, Anomaly-based NIDS (A-NIDS) is more efficient than Signature-based IDSs as they can detect new unseen attacks by creating a profile or a model after observing several examples of normal behavior [2,5]. As a result, there has been a plethora of research attempting to design an A-NIDS using a variety of techniques [5,6,15]. However, A-NIDSs suffer from performance issues such as low accuracy and a high false alarm rate compared to Signature-IDSs [5,14]. Therefore, designing an A-NIDS with a low false alarm rate and high accuracy is a challenging task which attracted extensive research in the literature [5].

In area of A-NIDS, various ML techniques have been used and can be categorized into two categories: unsupervised (clustering [1,4]) and supervised (classification [13,22,28]). Supervised techniques often show superior performance compared to unsupervised approaches, since the latter are generally sensitive to the initial parameters (e.g. number of clusters, distance metric) and the nature of data at hand. The use of unsupervised learning was motivated by the lack of hand-labeled data and the high cost of doing it manually, hence, the recourse to semi-supervised learning [11,23] as it reduces the cost of manually labeling the data. However, the obtained results remained below what supervised techniques achieved.

In this work, we conduct an exprimental evaluation using the UNSW-NB15 benchmark dataset [16] leveraging both traditional ML approaches and DL such as ANN. We have shown that the DL approach appears to be significantly superior to shallow learning because of its ability to learn a useful deep representation of the data, which is demonstrated by projecting the last hidden layer of our ANN architecture using the UMAP algorithm.

2 Related Work

The development of A-NIDS based on ML requires large and comprehensive datasets. NSL-KDD, a refined version of KDDCUP'99 [24] is one of the most widely used datasets in the literature for the evaluation of these systems. However, in [16,17], it has been stated that NSL-KDD suffers from several issues, such as the lack of modern low-footprint attacks and the smaller number of attack types compared to existing real-life attacks. To overcome these limitations, two new benchmark datasets have been generated, UNSW-NB15 [16] and CSE-CIC-IDS2018 [20]. In the latter, a train and test sets are not configured and provided to the community to allow a direct comparison between proposals, as is the case with UNSW-NB15. Furthermore, simple classifiers such as DT perform well in CSE-CIC-IDS2018, which means that it is less difficult and complex compared to the UNSW-NB15 dataset.

In [10], a performance analysis of the NSL-KDD benchmark data set has been performed using ANN in binary and multiclass classification with 81.2%

and 79.9% detection accuracies, respectively. More sophisticated deep learning algorithms were explored in the literature, such as Autoencoder (AE), and Recurrent Neural Network (RNN). The authors of [28] and [22] proposed the use of RNN by considering samples from the KDD dataset as a sequence. We know from [24] that individual training samples such as bidirectionnel flows are not necessarily related (i.e., they do not come from the same sequence). Therefore, the use of a sequence model, in this case, is not justified. In [17], an evaluation of the performance of A-NIDS using several classifiers such as DT, LR, Naive Bayes (NB), ANN, and Expectation Maximization Clustering (EMC) has been carried out using UNSW-NB15 dataset. The results show that the DT classifier achieves the highest accuracy with 85.56% and the lowest false alarm rate with 15.78%.

In [3], in addition to the hand-crafted header and statistical features extracted from the raw traffic, a new set of features has been computed using the proposed supervised Adaptive Clustering (AC) that are identical to the data samples (traffic flow) belonging to the same traffic class. Despite the good performance claimed by the authors, it remains an expensive architecture because of its complexity that rises exponentially as the number of classes increases (i.e. for n classes $n \times 2$ ANN architecture is used, each of which has its own set of weights and biases). The performance of the proposed technique has been evaluated using five synthetic data sets and three public benchmarks (KDD Cup'99 [9], ISCX-IDS 2012 [21] and CSE-CIC-IDS 2018 [20]). The authors claimed a 100% accuracy and 0% FAR in both binary and multiclass classification. However, the reproducibility of the results has not been achieved using the same settings. Furthermore, the authors did not test their solution on the UNSW-NB15 dataset, as it is widely used in the literature.

In [11], the authors proposed a semi-supervised technique that requires only 10 labeled data per flow type and relies on Generative Adversarial Networks (GANs) [7] to generate more data synthetically and reduce the cost manually labeling the data. In their proposal, they used a semi-supervised GAN (SGAN) [18,19] and achieved an accuracy of 88.7%.

In [13], the authors tried to address the problem of low performance of a standalone classification model by leveraging the use of Ensemble Learning (EL) in CPS (Cyber-Physical Systems) IDS. They evaluated three known EL techniques, such as aggregation, boosting, and stacking [29]. Throughout the experience, they observed that the ensemble models performed much better than the standalone models. When it comes to EL techniques, they showed that the stacking technique outperforms the others with an F1 score of 0.503 for the small imbalanced test set and a 0.996 for the large balanced test set. Although EL methods achieve good results, they have several drawbacks caused by their higher cost of creation, training, and deployment.

In [23], the authors proposed a semi-supervised approach using AEs to overcome the problems of supervised learning techniques such as the difficulty of detecting novel attack types and the need for large amount of labeled data (particularly the abnormal traffic) in the supervised training process. The learning

process consists in training an AE to reconstruct only normal samples by minimizing the reconstruction error of the model's input samples. When an abnormal sample is passed to the model, its reconstructed error would be higher compared to normal samples because it has not been seen in the training phase. Although the authors obtained encouraging results, their models cannot be generalized to multi-class anomaly detection and have not been tested in recent databases.

In [8], the authors proposed an early detection approach to address the supposed issues related to the detection of pos-tmortem attacks. Typically, the existing approaches for A-NIDS using ML take the full biflow packets (i.e. until the biflow ends) to extract the handcrafted features to train and test ML models. This approach is considered by the authors as weak defense strategy since it can not detect the attack on its early stage. They have experimented with a variable number of packets that first arrive in which belong to the same biflow using different ML algorithms. In their results, they showed that the use of RF classifier with as few as the first 10 packets allows to obtain a good performance-complexity trade-off. However, it should be noted that no comparison has been done with other existing A-NIDSs.

3 Dataset Description

The design of A-NIDS based on ML is heavily dependent on the availability of datasets that represent a wide area of Internet traffic, including modern and sophisticated attacks. There have been a lot of efforts in the literature to generate datasets. One of the earliest dataset released publicly to the community is NSL-KDD [24]. However, this dataset was collected more than decades ago and does not represent current normal and traffic patterns in the real world.

Several datasets have been released since then [16,20] in order to provide a modern and more inclusive view of Internet traffic.

In this work, we consider the UNSW-NB15 [16] in which lower performance has been obtained in the community compared to other datasets (as there is room for improvement) and the availability of training and testing sets that have been defined by the authors of the dataset so our work can be compared with other according to the same benchmarks defined by the authors.

The UNSW-NB15 dataset was created using the IXIA PerfectStorm tool [25] using the testbed configuration described in [16] to generate a mixture of modern real normal as well as synthetic attack traffic.

In this dataset, the packets belonging to the same bidirectional flow collected from the original captured raw data are grouped together and used to handcraft features about the biflow with the help of tools such as *Argus, Bro-IDS, and custom scripts*. These features are categorized into five groups: *Flow, Content, Time, Basic Features, and Generated Features which contains other features*. These handcrafted features are used to form a record in the constructed dataset.

Every record in the dataset is labeled as normal, or malicious in the case of binary labaling or into several types of malicious attacks in the case of multi-class labeling as can be shown in Table 1 which lists 9 attack categories of

malicious biflows (for more details about attack types, see the original paper of the dataset [16]).

The entire UNSW-NB15 dataset has a little over 2.5 million records (2,540,044 records). However, the authors deliberately configured a partition of this dataset as a training and testing set for a fair apple-to-apple future comparison, which contains 175341 and 82332 records, respectively. In Table 1, we show an illustration of the distribution of normal and malicious samples in the train and test sets.

Table 1. The number of records and the percentage of each class category in the USNW-NB15 train and test sets

(a) Classes	(b) Train set		(c) Test Set	
Category	**Nbr of records**	**Percentage**	**Nbr of records**	**Percentage**
Normal	56000	31.93%	37000	44.93%
Generic	40000	22.8%	18871	22.92%
Exploits	33393	19%	11132	13.52%
Fuzzers	18184	10.3%	6062	7.36%
DoS	12264	7%	4089	4.96%
Reconnaissance	10491	6%	3496	4.26%
Analysis	2000	1.14%	677	0.85%
Backdoor	1746	1%	583	0.70%
Shellcode	1133	0.6%	378	0.45%
Worms	130	0.07%	44	0.05%

4 Experimental Design

4.1 Dataset Pre-processing

Features Transformation: Among the 49 features of the data set, there are four categorical features such as *protocol*, *service*, *state*, and *attack-category*. Thus, a conversion of categorical features to numerical ones is required using one of the existing techniques such as the *One-Hot-Encoding*.

Features Scaling: It is a technique used to rescale the features of the data within a particular range, often between 0 and 1 or between -1 and 1. It improves the convergence speed of optimization algorithms such as gradient descent and its variants [26]. In this work, the Min-Max scaling technique has been considered to scale the data in the range of $[0, 1]$. Note that the feature scaling step is done using the train and test set separately to prevent data leakage from the test to the train set.

4.2 Evaluation Metrics

Generally, in IDS literature, two primary metrics are used to measure and evaluate the performance of the proposed solutions, such as the Accuracy and the False Alarm Rate (FAR). Accuracy is defined as the ratio of correctly classified samples to the total number of instances. The FAR is the average ratio of erroneously misclassified samples to correctly classified ones, which are calculated using the formula (1) and (2) respectively.

TP (True Positive): The number of malicious instances is correctly classified as malicious. TN (True Negative): The number of normal instances is correctly classified as normal. FP (False Positive): The number of normal instances is incorrectly classified as malicious. FN (False Negative): The number of malicious instances is incorrectly classified as normal.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$FNR = \frac{FN}{FN + TP} FPR = \frac{FP}{FP + TN} FAR = \frac{FPR + FNR}{2} \tag{2}$$

In addition to the two important metrics mentioned above, there are two other metrics that help further evaluate the proposed solutions, namely Precision and Recall. Precision is the ratio of correctly classified malicious samples to all samples classified as malicious, as illustrated in the formula (3). The recall as shown in (4) is the ratio of correctly classified malicious samples to the number of all malicious samples.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

5 Results and Discussion

This section is organized into two parts: in the first part we show the empirical results of binary classification, and in the second part we show the results of multi-class classification experiments using classical ML techniques and the ANN. All experiments were implemented using Python 3.7.0. Pandas and Numpy libraries were used for data manipulation. Sklearn version 0.24 and Keras version 2.4.3 with Tensorflow back-end version 2.4.0 for the implementation of the classical ML algorithms (LR, DT, RF) and DL (ANN), respectively.

For each part, after the preprocessing phase (see Sect. 4.1), and performing feature scaling in the interval [0,1] and feature transformation according to the one-hot technique, we obtained a new dataset with 196 normalized numerical features.

Table 2. ANN configuration in binary and multi-class classification

Hyper-parameters	Binary settings	Multi-class settings
Hidden layers	2	1
Neurons on hidden layer 1	128	64
Neurons on hidden layer 2	64	None
Neurons on output layer	1	10
Activation function in the hidden layers	Sigmoid	ReLu
Activation function in the output layer	Sigmoid	Softmax
Optimizer	RMSprop	RMSprop
Learning rate	0.01	0.001
Batch size	140	250
Epochs	40	100
Loss Function	Cross entropy	Cross entropy
Regularizer	L2 = 0.001	None
Kernel-initializer	Glorot uniform	Variance scaling

For training and evaluation, we use the official UNSW-NB15 benchmark train and test sets to allow a direct comparison with future research.

$$\mathcal{L}_{CE} = \begin{cases} -\frac{1}{M}\sum_{i=1}^{M}(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)), & \text{for Binary} \\ \\ -\frac{1}{M}\sum_{i=1}^{M}\sum_{c=1}^{C} y_{ic}\log(\hat{y}_{ic}), & \text{for Multi-class} \end{cases} \tag{5}$$

Table 3. Results of our models in the test set

(a) Binary results

classifier \ metric	Accuracy	Precision	Recall	FAR
LR	80.61%	74.94%	97.32%	21.21%
DT	86.13%	82.34%	95.22%	14.89%
RF	87.57%	84.74%	96.63%	12.33%
ANN	**88.77%**	**84.88%**	**96.86%**	**12.13%**

(b) Multi-class results

classifier \ metric	Accuracy
LR	69.14%
DT	72.76%
RF	70.47%
ANN	**73.13%**

5.1 Binary Classification

In this work, the classical ML classifiers were trained using the default hyper-parameters except for the LR which has been trained using the Limited-memory BFGS solver and a maximum iteration number of 500, and RF with a number of estimators (trees) equal to 200.

After experimenting with different hyper-parameters on the ANN classifier using the grid search technique, we end up with the binary settings shown in

Table 2. The network architecture has been selected to be 4 layers, including 2 hidden layers, each of which contains 128 and 64 artificial neurons, respectively. In the hidden and output layer, the activation function *sigmoid* is used. The optimization algorithm *RMSprob* has been chosen to minimize the Cross-Entropy loss function shown in formula (5), which differs from binary classification to multiclass.

The detailed results for binary classification indicate that ANN outperforms the other classifiers with 88.77% accuracy and 12.13% FAR as shown in Table 3a, while LR has the lowest performance in terms of accuracy and FAR with 80.61% and 21.21%, respectively.

The performance supremacy of ANN compared to traditional ML techniques is based on its ability to implicitly find other representations of data known as deep representation learning. To show how the ANN learns deep representation after the training phase, a UMAP 3-D projection has been used before training (i.e., before representation learning) with the initial set of features (194 features) and after training where the output of the second hidden layer neurons (64 features) is used as shown in Fig. 1a and 1b, respectively.

It is worth mentioning that almost all proposed ML techniques in the literature for A-NIDS have been trained and tested on randomly selected sets of UNSW-NB15 datasets despite the availability of a configuration from the authors to allow future research solutions to be compared in a fair manner (apple-to-apple comparison). We have used the architecture and the hyper-parameters of our best performing ANN model and we have trained and tested it by picking random samples from both sets. Repetition of the experiments several times has always shown an accuracy higher than 90% with a large variance (between [90%–98%] accuracy). This means that the proposed solutions can not be compared since their performance are directly affected by the chosen samples for train and test sets. For the reasons mentioned above, we are only able to invoke the experiments previously conducted on the same train and the test splits provided by the authors of the UNSW-NB15 data set and compare them with our results. As shown in Table 4, our ANN model clearly outperforms the performance of the proposed methods with an accuracy of 88.77% and a FAR of 12.13%.

Table 4. A comparison of our proposed models with other previous works in binary classification

Paper	Technique	Accuracy	FAR
R. Vinayakumar *et al.* [27]	ANN	76.5%	//
T. Kim *et al.* [12]	Encoding + 2D CNN	80%	//
N. Moustafa *et al.* [17]	ANN	81.34%	21.13%
N. Moustafa *et al.* [17]	Decision Tree	85.56%	15.78%
Ours	**ANN**	**88.77%**	**12.13%**

5.2 Multi-class Classification

ML-based techniques have been implemented using the by default hyperparameters except LR where the solver, maximum iteration, and multi-class hyperparameters have been set to Limited-memory BFGS solver, 1100 and multinomial, respectively. Moreover, the RF classifier has been implemented with 30 estimators.

The ANN classifier has been implemented with the multi-class settings shown in Table 2 after experimenting with different hyper-parameters using the grid search technique. The classifier have been trained using 1 hidden layer with 64 neurons and the *ReLu* as an activation function. In the output layer, the number of neurons was set to 10 equally to the number of existing classes and the *softmax* function as an activation function that normalizes the input into a probability distribution.

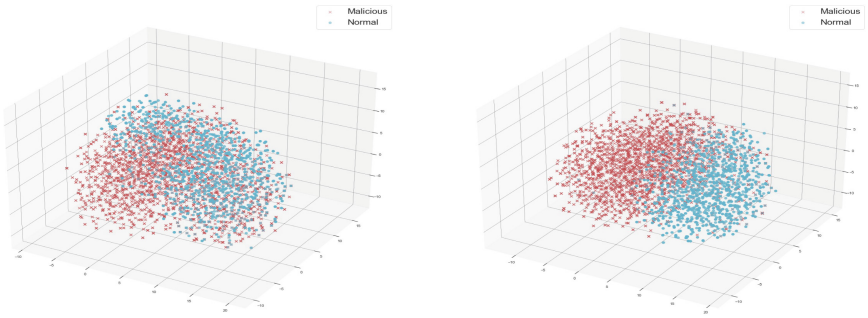
The computation of the evaluation metrics on multi-label classification is a bit different from that on binary classification. Accuracy is calculated by summing the correctly classified samples for each class divided by the sum of all elements of the confusion matrix. However, precision, recall, and false alarm rate can not be computed regarding the performance of the overall model as in binary classification. As a result, the confusion matrix (See Fig. 3) is given to assess the classifier in depth.

The experimental results shown in Table 3b indicate that the accuracy of the ANN classifier (73.13%) is better than that of the classical ML techniques. In addition, the accuracy of the classifiers on multi-class classification has declined compared to binary classification.

As the confusion matrix (see Fig. 3) indicates, our model performs much better in some classes than in others. For instance, the proportion of correctly predicted samples from the Generic class is 96% and 79% for the Normal class while 2%, 19%, and 1% for Worms, Shellcode, and Backdoor, respectively.

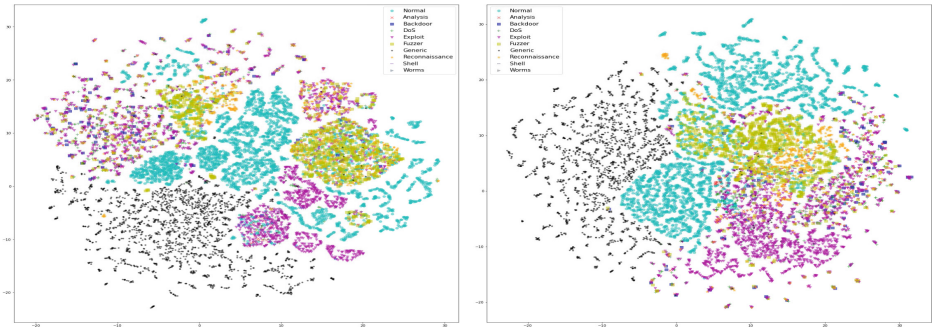
The primary reason behind this phenomenon is the unbalanced nature of the observations on the training set of UNSW-NB15 dataset. Normal and Generic categories make 31.93% and 22.80% of trainig set, whereas Worms, Shellcode, and Backdoor combined make only 1.67% as mentioned on the Table 1 which is a huge difference on the distribution. Therefore, the unbalanced nature of the data has led the model to learn well the patterns of more populated classes compared to less populated in which has resulted to a decrease of the performance in multi-class classification.

Finally, a 2-D projection of 1000 random samples from the training set is conducted before and after training the ANN classifier (Fig. 2a, 2b respectively) to show that the learned representation during the training process using ANN classifier are much more helpful in distinguishing between biflow classes.



(a) Before Training (b) After Training

Fig. 1. A 3-D projection of 10000 randomly selected samples before training (a) and after training (b) by selecting the second layer output of our ANN architecutre in binary classification using UMAP algorithm.



(a) Before Training (b) After Training

Fig. 2. A 3-D projection of 10000 randomly selected samples before training (a) and after training (b) by selecting the second layer output of our ANN architecutre in multi-class classification using UMAP algorithm.

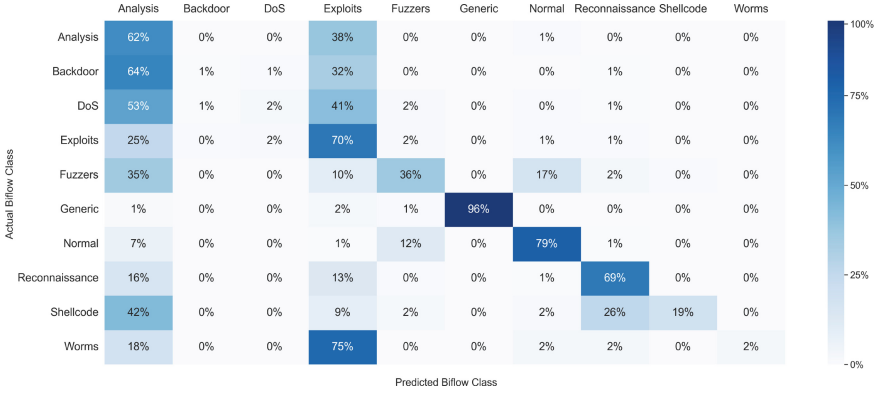


Fig. 3. The Confusion Matrix of our ANN model in multi-class classification

6 Conclusions

In this work, we evaluated different ML and DL techniques for the developpement of A-NIDS using the official public benchmark UNSW-NB15 data set. The results show that the ANN model performs well on previously unseen biflow attacks with high accuracy and low FAR compared to the traditional ML techniques on both binary and multi-label classification. Furthermore, a visualization technique named UMAP has been used to show the core advantage of ANN over the traditional techniques in extracting useful representations from the data that ultimately improve the performance of the model in the detection of malicious biflows, which makes it eligible with its variants to improve the performance of A-NIDS in future works. Moreover, we have empirically shown that the imbalanced nature of the data set greatly affects the performance of the model by learning well on the populated classes compared to the less populated ones. In future research, we believe that reformulating the problem of A-NIDS is required to leverage some more sophisticated DL approaches. Moreover, we aim to apply deep generative models techniques such as GANs (Generative Adversarial Networks) and VAEs (Varational Auto Encoders) to address the issue of data imbalance by generating synthetic data samples for the classes that have insufficient samples that would theoretically improve the performance of the classifier.

References

1. Bigdeli, E., Mohammadi, M., Raahemi, B., Matwin, S.: Incremental anomaly detection using two-layer cluster-based structure. Inf. Sci. **429**, 315–331 (2018)
2. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutor. **18**(2), 1153–1176 (2016)
3. Diallo, A.F., Patras, P.: Adaptive clustering-based malicious traffic classification at the network edge. In: IEEE INFOCOM, pp. 1–10. IEEE (2021)
4. Dromard, J., Roudière, G., Owezarski, P.: Online and scalable unsupervised network anomaly detection method. IEEE Trans. Netw. Serv. Manag. **14**(1), 34–47 (2017)
5. Fernandes, G., Rodrigues, J.J.P.C., Carvalho, L.F., Al-Muhtadi, J.F., Proença, M.L.: A comprehensive survey on network anomaly detection. Telecommun. Syst. **70**(3), 447–489 (2019). https://doi.org/10.1007/s11235-018-0475-8
6. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**(1–2), 18–28 (2009)
7. Goodfellow, I.J., et al.: Generative adversarial networks (2014)
8. Guarino, I., Bovenzi, G., Di Monda, D., Aceto, G., Ciuonzo, D., Pescapé, A.: On the use of machine learning approaches for the early classification in network intrusion detection. In: IEEE M&N, pp. 1–6 (2022)
9. Hettich, S., Bay, S.D.: KDD Cup 1999 Data
10. Ingre, B., Yadav, A.: Performance analysis of NSL-KDD dataset using ANN. IN: SPACES, pp. 92–96 (2015)

11. Jeong, H., Yu, J., Lee, W.: Poster abstract: a semi-supervised approach for network intrusion detection using generative adversarial networks. IEEE Infocom, pp. 31–32 (2021)
12. Kim, T., Suh, S.C., Kim, H., Kim, J., Kim, J.: An encoding technique for CNN-based network anomaly detection. In: Big Data, pp. 2960–2965 (2019)
13. Li, H., Chasaki, D.: Ensemble machine learning for intrusion detection in cyber-physical systems. In: INFOCOM WKSHPS, pp. 12–13. IEEE (2021)
14. Liao, H.J., Richard Lin, C.H., Lin, Y.C., Tung, K.Y.: Intrusion detection system: a comprehensive review. J. Netw. Comput. Appl. **1**, 16–24 (2013)
15. Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S.: A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun. Surv. Tutor. **21**(1), 686–728 (2019)
16. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: MilCIS, pp. 1–6. IEEE (2015)
17. Moustafa, N., Slay, J.: The evaluation of Network Anomaly Detection Systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf. Secur. J. **25**(1–3), 18–31 (2016)
18. Odena, A.: Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:1606.01583 (2016)
19. Salimans, T., et al.: Improved techniques for training GANs, vol. 29. Curran Associates, Inc. (2016)
20. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP 2018 (Cic), pp. 108–116 (2018)
21. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput. Secur. **3**, 357–374 (2012)
22. Staudemeyer, R.C.: Applying long short-term memory recurrent neural networks to intrusion detection. S. Afr. Comput. J. **56**(56), 136–154 (2015)
23. Švihrová, R., Lettner, C.: A semi-supervised approach for network intrusion detection. In: ACM International Conference Proceeding Series (2020)
24. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009)
25. Tool, I.P.: https://www.keysight.com/fr/en/products/network-test/network-test-hardware/perfectstorm.html
26. Tsakalidis, S., Doumpiotis, V., Byrne, W.: Discriminative linear transforms for feature normalization and speaker adaptation in HMM estimation. IEEE Trans. Speech Audio Process. **13**(3), 367–376 (2005)
27. Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S.: Deep learning approach for intelligent intrusion detection system. IEEE Access **7**, 41525–41550 (2019)
28. Yin, C., Zhu, Y., Fei, J., He, X.: A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access **5**, 21954–21961 (2017)
29. Zhang, C., Ma, Y.: Ensemble Machine Learning. Springer, New York (2012). https://doi.org/10.1007/978-1-4419-9326-7